Name:

ID:

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 6 – Due Fri Feb 28 11:55pm** **Spring 2020, CU-Boulder**

---

*Advice 1*: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

*Advice 2*: Informal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

**Instructions for submitting your solutions**:

- All submissions must be easily legible.
- You should submit your work through the **class Canvas page** only.
- You may not need a full page for your solutions; pagebreaks are there to help Gradescope automatically find where each problem is. Even if you do not attempt every problem, please allot at least as many pages per problem (or subproblem) as are allotted in this template.
- For drawing graphs, you may include scans of hand-drawn graphs into your PDF file. **However, the rest of your solution (including the explanation of the graph) must be typed. If your words are not typed, you will get a 0 for that part of the question.**

Quicklinks: 1 2 3 4a 4b 4c

---

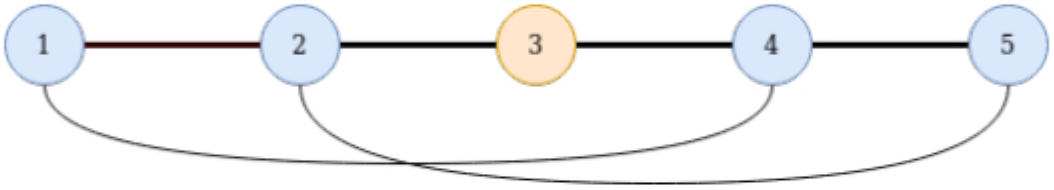
Name:

ID:

**CSCI 3104, Algorithms** **Profs. Chen & Grochow**

**Problem Set 6 – Due Fri Feb 28 11:55pm** **Spring 2020, CU-Boulder**

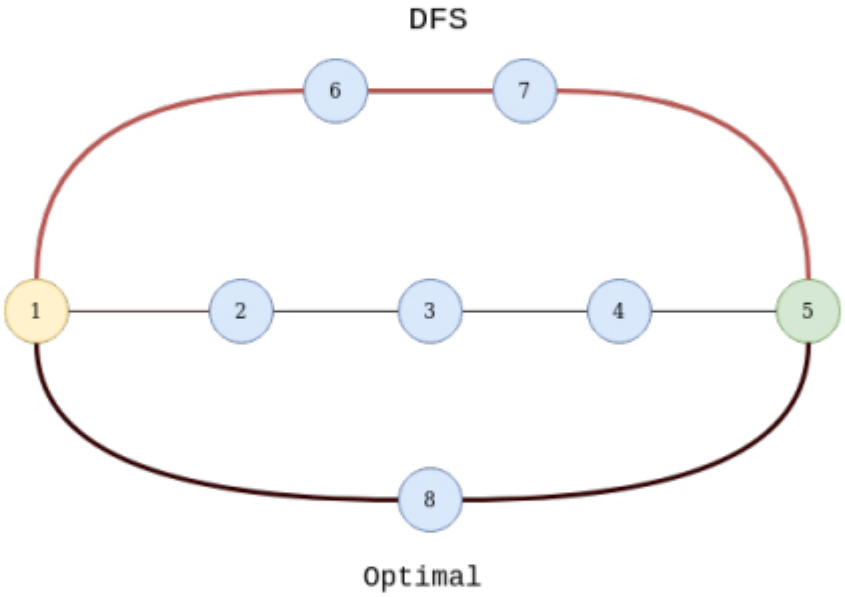
1. Give an example of a (simple, undirected) graph $G = (V, E)$, a start vertex $s \in V$ and a set of tree edges $E_T \subseteq E$ such that for each vertex $v \in V$, the unique path in the graph $(V, E_T)$ from $s$ to $v$ is a shortest path in $G$, yet the set of edges $E_T$ cannot be produced by running a breadth-first search on $G$, no matter how the vertices are ordered. Include an explanation of why your example satisfies the requirements.




Starting at vertex 3, BFS will add nodes 2 and 4 to its queue on the first iteration, adding each of the edges. If it pops 2 off the queue, it will then add 1 and 5, using one bold edge, and one of the long rounded edges, which is not part of $E_T$. On the other hand, if it pops 4 off the queue first, then it will add 1 and 5, again using one bold edge, and one long rounded edge, which is not part of $E_T$. Either way, the resulting BFS tree is not equal to the one consisting of the bold edges.

Finally, it is not hard to see that the bold edges form a SSSP tree from 3: 2 and 4 are at distance 1 from 3 in $G$, and also in the tree, and 1 and 5 are at distance 2 from 3 in $G$, and also in the tree.

2. A *simple* $s \to t$ *path* in a graph $G$ is a path in $G$ starting at $s$, ending at $t$, and never visiting the same vertex twice. Give an example graph (simple, undirected, unweighted) $G = (V, E)$ and vertices $s, t \in V$ such that DFS finds a path from $s$ to $t$ which is neither a shortest path nor a longest simple path. Detail the execution of DFS (list the contents of the queue at each step, and which vertex it pops off the queue), show the final $s \to t$ path it finds, show a shorter $s \to t$ path, and a longer simple $s \to t$ path.




Let $s = 1$ and $t = 5$ Here we assume DFS uses some arbitrary ordering (not necessarily determined by node names) which causes it to follow the path 1-6-7-5. This path is neither the shortest nor the longest path in the graph. We have a shorter path 1-8-5 and a longer path 1-2-3-4-5, both of which would be possibly but not necessarily discovered by DFS, depending on the order in which it selects nodes.